Prosodic inference with the ZDATR default inference engine

Dafydd Gibbon

Universität Bielefeld, Germany
gibbon@uni-bielefeld.de

ABSTRACT

Linguistic rules, constraints, processes, and other instruments for expressing generalisations tend to be formulated as isolated statements whose relationship to each other remains unclear and is rarely formulated explicitly (with notable exceptions like the stress cycle, or layers of morphological rule organisation). The present contribution demonstrates a default logic technique using the DATR knowledge representation language, with implementation in ZDATR, for ensuring the connectedness of the theory, in application to several non-trivial prosodic problems in English, German and Niger-Congo languages including /Yacouba, Kikuyu, Tem and Baule.

1 Default inference in prosody

1.1 Objectives

In contemporary discussions on prosody, as in phonology and phonetics in general, the instruments for expressing generalisations are referred to as *rules*, *processes*, *constraints*, and *principles* as components of *theories*, *grammars*, *automata* and related constructs, and they vary in different approaches. However, it is in general very difficult or even impossible to tell how the different rules and constraints relate to each other. The rules, constraints etc. are formulated in relative isolation from each other, and related by means of highly specific ordering rules. Notable exceptions to this *ad hoc* state of affairs (leaving aside the numerous empirical problems) are the morphophonological cycles in the Generative Phonology, Lexical Phonology and Metrical Phonology of the 1960s, 1970s and 1980s. Rules which are formulated in isolation and without well-motivated interconnections infringe a well-tried heuristic principle in formal linguistics, that 'une langue est un système où tout se tient', i.e. a language is a fully connected system.

A remedy for this situation is to conceive of rules as axioms, from which theorems may be derived (inferred, predicted) by highly general logical principles of inference which are not motivated *ad hoc* by particular descriptive problems, and to demonstrate how theorems may be derived from these axioms

by means of an automatic theorem-prover; connectivity of the inference steps may be demonstrated by the use of inheritance graphs.

The theorems are tested against models of reality which can in turn be constructed from induction over intuitive observations, production and perception experiments, and measurements of corpus data. The present contribution is concerned with a direct application of default logic for reconstructing linguistic generalisations, concentrating more on formal issues rather than on detailed empirical grounding.[1]

1.2 ZDATR: a theorem-prover as a default inference engine

For present purposes, the default logic formalism DATR[2] is used, for which a number of theorem provers have been implemented[3]. For conformity with the standard RFC specification (Evans & Gazdar 1998) the ZDATR[4] inference engine is used here. DATR is a dedicated formalism which is used to define theories as sets of axioms from which theorems are derived by means of defeasible (overridable) and non-defeasible (standard) inference. The theorem strings are constructed compositionally by concatenation. DATR theories are often modelled either as inheritance hierarchies or as formal automata, depending on whether the focus is on deduction or on string composition. A typical linguistic application of DATR is to describe an 'inheritance lexicon' in which inflected and derived forms of words are constructed from a stem and an

---

1 I have profited greatly from discussions about many technical, phonetic and prosodic issues with Stefan Grocholowski over the years in the context of ongoing research cooperation with Grażyna Demenko. These discussions have encourage me to contribute an account of a specific formal technique of handling formal issues in inference with prosodic generalisations of a wide variety of types. I am particularly grateful to Gerald Gazdar, Doris Bleiching, Katja Jasinskaja and Sabine Reinhard for extensive discussions on these issues.

2 The name "DATR", pronounced [ˈdætə] is not an acronym. It was coined by Roger Evans and Gerals Gazdar by analogy with "PATR II", a parser for attribute-value systems.

3 There have been many implementations of DATR inference engines, of which Sussex DATR (Evans & Gazdar 1996), QDATR (James Kilbury), KATR (Finkel & Stump n.d.), ZDATR (Gibbon & Strokin 1998) are currently available as open source software. KATR contains an additional abbreviatory notation which expresses DATR paths as sets, thus capturing arbitrary orderings of the subset of DATR paths which contain no atom doublets.

4 Unlike previous implementations in Prolog, Scheme and Java, which have various non-standard idiosyncrasies, ZDATR is an efficient implementation in C which is fully conformant with the DATR version specified in RFC 2.0 (Evans & Gazdar 1998). ZDATR has the reputation of being the industry strength standard implementation of DATR, and is intended for command line use in large-scale industrial strength pipelined applications as well as for small prototype development. The name "ZDATR", pronounced [ˈzdɑtr], is not an acronym; the "Z" (as the last letter in the alphabet) is intended to suggest that ZDATR is the definitive RFC 2.0 version of DATR. ZDATR sources, binaries and documentation are available at the following address: http://coral.spectrum.uni-bielefeld.de/DATR/

implication hierarchy of inflection classes, and where irregularities override the defeasible general regular generalisations.

The present approach uses both the inheritance and the composition paradigms to interpret inference of prosodic patterns in DATR. Similar inheritance techTemniques to those on which DATR is based have been well-known in the object-oriented programming paradigm since the 1970s and are standardly used in modern software development. DATR shares the heuristic utility of this paradigm for practical modelling and development purposes.

1.3 Descriptive problems for prosodic inference

The specific prosodic modelling problems dealt with here are taken on the one hand from English and German, i.e. stress and intonation languages, and on the other hand from the lexico-syntactic tone languages of West Africa. The problems covered are stress assignment in German word formation, markedness relations between nuclear tones in English intonation, lexical prosody in Dan/Yacouba, a Mande language of Western Ivory Coast, tonal displacement in Kikuyu, a Bantu language of East Africa, and the phonetic realisation of terraced tone sandhi in two-tone Niger-Congo languages such as Tem (Togo) and Baule (Côte d'Ivoire).

2 The default inference formalism DATR

2.1 DATR inference engine definitions

Almost every article ever published on inference with DATR has included a more or less useful informal introduction to the formalism (see bibliography, also a wide selection on the internet), so the introduction in the present context can be strictly technical, and limited to a set of definitions, and a simple intuitively comprehensible example of inference with the ZDATR engine. The definitions are kept to fairly standard notations, including a small subset of regular expressions.

A DATR query process is a triple <*E*, *T*, *Q*>, where
E is an inference engine implementing
  D: a default matching rule
  S: a rule of suffix attachment
  I: a set of 7 rules of inference (defined below)
  R: a register triple consisting of
    GC: a global context register, values accessible any time after assignment
    LC: a local context register, values accessible only at the current node
    RS: a suffix register
T is a theory consisting of
  A: a set of atoms (character strings, single-quoted if with special characters)
  N: a set of nodes, each paired with
    E: a set (ending in ".") of equations, each consisting of *LHS* == *RHS*
      for
      LHS: a delimited path (sequence) of atoms, i.e. < `atom*` >
      RHS: a recursive sequence (*atom*|*N*|*RP*|*NRP*|"*N*"|"*RP*"|"*NRP*")*,

for
N: node
RP: recursive path <(*atom*|*N*|*RP*|*NRP*|"*N*"|"*RP*"|"*NRP*")*>,
NRP: pair of a node and a recursively defined path, and
N, *RP*, *NRP* are evaluated in the local context
"*N*", "*RP*", "*NRP*" are evaluated in the global context
Q a query, concatenation of a node, ":", and an atom path: *Node:*< a*tom** >

The following is a typical DATR theory which illustrates most of these structures and models a tiny lexicon in a style which has been used in speech technology projects:

```
Cat:                  % Abstract lexical lemma node
  <> == Noun          % Default inference from Noun node
  <phon> == 'kat'     % Phonemic representation in SAMPA
  <sem> == feline.    % Simple semantic representation
Dog:                  % Abstract lexical lemma node
  <> == Noun          % Default inference from Noun
  <phon> == 'dog'     % Phonemic representation in SAMPA
  <sem> == canine.    % Simple semantic representation
Noun:                 % Generalisation node for nouns
  <> == Word          % Default inference from Word node
  <syn> == noun.      % Generalised property of nouns
Word:                 % Generalisation node for words
  <> ==               % Default value is null
  <entry> == [ "<phon>" "<sem>" "<syn>" ]. % Template
```

In this axiom set for a lexicon theory, the microstructure of the entries Cat and Dog is underspecified and missing information is inferred from information at more abstract nodes shared by several more specific nodes. In this theory, this applies to parts of speech (here nouns) and a vector template for the lexical entry of a word. The following theorems can be inferred from the lexicon theory (phonemic representations are in the SAMPA alphabet):

```
Cat:< entry > = [ kat feline noun ] .
Cat:< phon > = kat .
Cat:< sem > = feline .
Cat:< syn > = noun .
Dog:< entry > = [ dog canine noun ] .
Dog:< phon > = dog .
Dog:< sem > = canine .
Dog:< syn > = noun .
```

2.2 Recursive default inference

Inference proceeds in the following steps:

Initialisation:

1. The global context *GC* is initialised to the query *Q*.
2. The local context *LC* is initialised to the global context *GC*.

Matching:

3. The local context *LC* is matched to
    1. a node *N* in the theory *T* which matches the node in *LC*,
    2. the longest *LHS* at the node *N* which is a prefix of (or equal to) the path in *LC*.

4. The suffix register *RS* is initialised to the non-matched suffix of the matched query *Q*.

Evaluation (deduction):

5. Each element of the *RHS* is evaluated and the values are concatenated, whereby
    1. atoms evaluate to themselves (Rule I),
    2. *GNRP*, *GN*, *GRP* and re-initialise the *GC* and *LC* registers and evaluate recursively (Rules II, III, IV),
    3. *LNRP, LN*, *LRP* reinitialise the *LC* register and evaluate recursively (Rules V, VI, VII),
    4. the value of the suffix register *RS* is appended to each *RP* in *GC* and *LP*.

Recursion:

6. Matching and Evaluation are recursively applied to *GC* and *LC*.

The essential feature for the default logic interpretation of DATR notation is the "longest path match" criterion expressed in step 3.2, which explicates the principle that a more general value (corresponding to a shorter match) applies *unless* a more specific criterion (corresponding to a longer match) applies.

The following theorems are deduced from the queries listed above by means of the inference procedure outlined here:

```
Cat:< entry > = [ k{t feline noun ] .
Cat:< phon > = k{t .
Cat:< sem > = feline .
Cat:< syn > = noun .
Dog:< entry > = [ dOg canine noun ] .
Dog:< phon > = dOg .
Dog:< sem > = canine .
Dog:< syn > = noun .
```

To gain an intuitive understanding of the details of this process, a trace of the inference steps for a simple theorem (generated with the ZDATR inference engine) is shown here:

Query:

```
Cat:< phon >
```

Theorem:

```
Cat:< phon > = k{t .
```

Inference:

```
#1=> Cat:<sem>
=0,0,0> LOCAL Cat:< sem > == feline
        GLOBAL Cat:< sem >
RULE I.(ATOM)
feline
[Query 1 (1 Inferences)] Cat:< sem >
 = feline .
```

The trace in ZDATR interactive mode shows the following, line by line:

1. First query in the interaction, with query `Cat:<sem>`
2. The series of zeroes shows the inference number and the depths of global and local recursion. The content of the LOCAL context register matches with the node-path pair `Cat:<sem>` in the equation with the atomic value `feline`, with no unmatched suffix part.

3. The GLOBAL context register contains the query node-path pair.
4. Inference rule I (of seven) applies: an atom on the RHS evaluates to itself.
5. The value of this equation thus evaluates to ‘feline’.
6. No recursion is necessary, and the entire theorem is inferred after only one inference.

The definitions already given should provide enough information to understand the derivation of the following more complex theorem:

```
#1=> Cat:<entry>
=0,0,0> LOCAL Cat:< || entry > == Noun
        GLOBAL Cat:< entry >
=1,0,0> LOCAL Noun:< || entry > == Word
        GLOBAL Cat:< entry >
=2,0,0> LOCAL Word:< entry > == [ "< phon >" "< sem >"
   "< syn >" ]
        GLOBAL Cat:< entry >
=3,0,1> LOCAL Cat:< phon > == k{t
        GLOBAL Cat:< phon >
=3,0,2> LOCAL Cat:< sem > == feline
        GLOBAL Cat:< sem >
=3,0,3> LOCAL Cat:< || syn > == Noun
        GLOBAL Cat:< syn >
=4,0,0> LOCAL Noun:< syn > == noun
        GLOBAL Cat:< syn >
[Query 1 (11 Inferences)] Cat:< entry >
 = [ k{t feline noun ] .
```

The prosodic inference cases discussed in the following sections of the present contribution follow exactly the same inference rules.

3 A selection of inference strategies

3.1 Overview: prosodic relations

Prosody is sometimes naively conceived as English-style intonation and stress patterns. But prosody has long been known to be far more complex than this, covering not only lexical stress and its realisation by phonetic prominence patterns, but also, in a wide range of languages, the complexities of lexical and morphosyntactic tone. The scope of prosody will not be discussed here but can be inferred from the examples discussed below.

Prosodic inference in one German and one English case will be treated, followed by a discussion of lexical and supralexical prosody in two Niger-Congo languages. Following this, tone sandhi inference will be discussed using a generic approach which applies to the majority of Niger-Congo two-tone languages and has also been used in speech synthesis tone generation.[5]

5 Documented versions of the theories discussed in this section are easily locatable via internet search or directly on the ZDATR web site: http://www.spectrum.uni-bielefeld.de/DATR/

3.2 Inference of stress patterns in German word-formation

The problem to be solved concerns the assignment of stress patterns in German compound words (Gibbon & Bleiching 1991; Bleiching 1992, 1994). The theory itself cannot be presented here as it contains several hundred morphological, phonological and prosodic equations. Consequently only a small selection of theorems and one simple derivation will be shown in order to illustrate the complexity of the problem.

It is not possible to provide a complete derivation here, as it involves over 100 inference steps. The simplex word "Zug" appears superficially to be simple, but when all the details of segments and syllable hierarchy are taken explicitly into account, inference for "Zug" alone turns out to be quite lengthy, involving 31 steps, as the following trace shows:

```
=0,0,0> LOCAL Zug:< || surf phon qlp > == N
        GLOBAL Zug:< surf phon qlp >
=1,0,0> LOCAL N:< surf phon qlp > == QLP:<  >
        GLOBAL Zug:< surf phon qlp >
=2,0,0> LOCAL QLP:< > == < qlp "< morph cat >" >
        GLOBAL Zug:< surf phon qlp >
=3,1,0> LOCAL Zug:< || morph cat > == N
        GLOBAL Zug:< morph cat >
=4,1,0> LOCAL N:< morph cat > == root
        GLOBAL Zug:< morph cat >
=3,0,0> LOCAL QLP:< qlp root > == Root:<  >
        GLOBAL Zug:< surf phon qlp >
=4,0,0> LOCAL Root:< > == [ Stress MP_Onset MP_Nucleus
  MP_Coda ]
        GLOBAL Zug:< surf phon qlp >
=5,0,1> LOCAL Stress:< > == "
        GLOBAL Zug:< surf phon qlp >
=5,0,2> LOCAL MP_Onset:< > == O_sib O_con O_ext
        GLOBAL Zug:< surf phon qlp >
=6,0,0> LOCAL O_sib:< > == "< surf phon ons sib >"
        GLOBAL Zug:< surf phon qlp >
=7,0,0> LOCAL Zug:< || surf phon ons sib > == N
        GLOBAL Zug:< surf phon ons sib >
=8,0,0> LOCAL N:< || surf phon ons sib > ==
        GLOBAL Zug:< surf phon ons sib >
=6,0,1> LOCAL O_con:< > == "< surf phon ons con >"
        GLOBAL Zug:< surf phon qlp >
=7,0,0> LOCAL Zug:< surf phon ons con > == ts
        GLOBAL Zug:< surf phon ons con >
=6,0,2> LOCAL O_ext:< > == "< surf phon ons ext >"
        GLOBAL Zug:< surf phon qlp >
=7,0,0> LOCAL Zug:< || surf phon ons ext > == N
        GLOBAL Zug:< surf phon ons ext >
=8,0,0> LOCAL N:< || surf phon ons ext > ==
        GLOBAL Zug:< surf phon ons ext >
=5,0,3> LOCAL MP_Nucleus:< > == N_vow N_ext
        GLOBAL Zug:< surf phon qlp >
```

```
=6,0,0> LOCAL N_vow:< > == "< surf phon nuc vow >"
        GLOBAL Zug:< surf phon qlp >
=7,0,0> LOCAL Zug:< surf phon nuc vow > == u
        GLOBAL Zug:< surf phon nuc vow >
=6,0,1> LOCAL N_ext:< > == "< surf phon nuc ext >"
        GLOBAL Zug:< surf phon qlp >
=7,0,0> LOCAL Zug:< surf phon nuc ext > == :
        GLOBAL Zug:< surf phon nuc ext >
=5,0,4> LOCAL MP_Coda:< > == C_sib C_con C_ext
        GLOBAL Zug:< surf phon qlp >
=6,0,0> LOCAL C_sib:< > == "< surf phon cod sib >"
        GLOBAL Zug:< surf phon qlp >
=7,0,0> LOCAL Zug:< || surf phon cod sib > == N
        GLOBAL Zug:< surf phon cod sib >
=8,0,0> LOCAL N:< || surf phon cod sib > ==
        GLOBAL Zug:< surf phon cod sib >
=6,0,1> LOCAL C_con:< > == "< surf phon cod con >"
        GLOBAL Zug:< surf phon qlp >
=7,0,0> LOCAL Zug:< surf phon cod con > == k
        GLOBAL Zug:< surf phon cod con >
=6,0,2> LOCAL C_ext:< > == "< surf phon cod ext >"
        GLOBAL Zug:< surf phon qlp >
=7,0,0> LOCAL Zug:< || surf phon cod ext > == N
        GLOBAL Zug:< surf phon cod ext >
=8,0,0> LOCAL N:< || surf phon cod ext > ==
        GLOBAL Zug:< surf phon cod ext >
[Query 1 (42 Inferences)] Zug:< surf phon qlp > =
  ["tsu:k].
```

The following theorems illustrate (without full traces) the compositional assignment of structure and stress to the simplex word “Zug” (train), the derived word “Verbindung” (connection) and the compound word “Zugverbindung” (train connection):

```
Zug:< surf phon qlp > = ["tsu:k].
Verbindung:< surf phon qlp > = [f@R]^["bInd]^[Ung].
Zugverbindung:< surf phon qlp > =
  ["tsu:k]^[f@R]^[%bInd]^[Ung].
```

Primary and secondary stress are indicated by “"” and “%” respectively. The path `<surf phon qlp>` is defined in Integrated Lexicon with Exceptions (ILEX) theory (Gibbon 1992), and indicates a hierarchical feature structure for the “quasi-linear precedence” (qlp, multi-tier) relation, the “phonological” (phon) option of the “surface” (surf) interpretation of an abstract lemma in the lexicon. In this theory, surface interpretation is opposed to semantic interpretation, phonological interpretation is opposed to graphemic interpretation in written text, and the quasi-linear precedence prosodic or autosegmental representation is opposed to the linear segmental representation of the segmental phonemes.

3.3 Markedness relations between nuclear tones in English intonation

The ILEX approach can be generalised to the semantic and phonetic interpretation of nuclear tone patterns in English. In these examples, the details of notation (e.g. levels vs. contours) are unimportant; the important issue is that the interpretation of intonation patterns can be treated as compositional. In the following theory, the contours *rise-fall*, '*calling*', *high*, *low*, *rise* and *fall* are treated (the 'calling' tone is used in vocative calls; cf. Gibbon 1976).

```
Rise_Fall:
  <> == Complex_Prosody
  <sem> == 'appraisive'
  <phon> == broad_bandwidth
  <const specifier> == "Rise:<>"
  <const head> == "Fall:<>".
Call_Contour:
  <> == ( "Chroma" Complex_Prosody )
  <sem> == 'call'
  <phon> == minor_third
  <category> == tonal_idiom
  <const specifier> == "High:<>"
  <const head> == "Mid:<>".
Rise:
  <> == Complex_Prosody
  <sem> == 'suspense'
  <phon> == open
  <const specifier> == "Low:<>"
  <const head> == "High:<>".
Fall:
  <> == Complex_Prosody
  <sem> == 'certainty'
  <phon> == closed
  <const specifier> == "High:<>"
  <const head> == "Low:<>".
High:
  <> == Prosody
  <category> == tonal_terminal
  <sem> == 'continue'
  <phon> == high.
Mid:
  <> == Prosody
  <sem> == 'hesitate'
  <phon> == mid.
Low:
  <> == Prosody
  <sem> == 'stop'
  <phon> == low.
Chroma:
  <> == Complex_Prosody
  <inter> == <>
  <sem> == phatic '*'
```

```
  <phon> == chroma '*' .
Complex_Prosody:
  <> == Prosody
  <category> == complex_tone
  <inter> == ( "<>" * "<const specifier inter>" '&'
   "<const head inter>" ) .
Prosody:
  <> == simple_tone
  <inter> == "<>"
  <category> ==  "<const head category>".
```

Figure 1 visualises the English nuclear tone system as a fully connected 'système où tout se tient' in a network diagramme interpreted according to the inheritance paradigm, in which all nodes are directly or indirectly connected to the others. The relations between the terminal nodes (i.e. the nodes which can be queried, are represented with continuous circles) and the abstract, general nodes (i.e. the nodes which cannot in general be queried sensibly, are represented by dotted circles). The global inference relations are represented by dotted lines, and the local inference relations are represented by continuous lines.

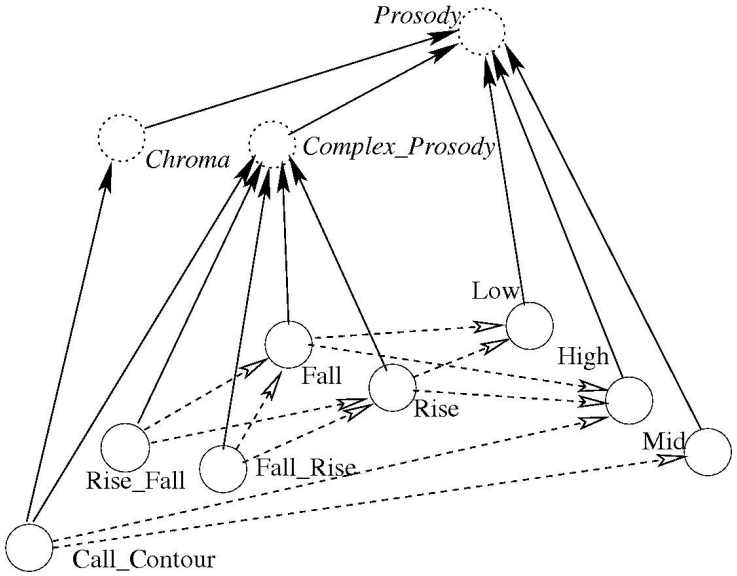

Figure 1: Compositional treatment of semantic and phonetic interpretation of nuclear tones in English.

Some of the inferences which can be drawn with this theory are as follows:

```
Call_Contour:< inter sem > = ( phatic * ( call *
  continue & hesitate ) ) .
Call_Contour:< inter phon > = ( chroma * ( minor_third *
  high & mid ) ) .
Fall:< inter sem > = ( certainty * continue & stop ) .
Fall:< inter phon > = ( closed * high & low ) .
```

```
High:< inter sem > = continue .
High:< inter phon > = high .
Low:< inter sem > = stop .
Low:< inter phon > = low .
Mid:< inter sem > = hesitate .
Mid:< inter phon > = mid .
Rise:< inter sem > = ( suspense * stop & continue ) .
Rise:< inter phon > = ( open * low & high ) .
Rise_Fall:< inter sem > = ( appraisive * ( suspense *
  stop & continue ) & ( certainty * continue &
  stop ) ) .
Rise_Fall:< inter phon > = ( broad_bandwidth * ( open *
  low & high ) & ( closed * high & low ) ) .
```

The semantic labels are informal, and will not be discussed further here.

3.4 Lexical prosody in Dan (Yacouba)

The ILEX approach is used to assign lexical tone in a theory (originally formulated in French, cf. Gibbon & Ahoua 1991) of the lexical structure of Dan (Yacouba, ISO 639.3: daf), a Mande language of Western Ivory Coast and Eastern Liberia. The following example of a lexical entry shows the hierarchical, immediate dominance (ID) structure of the phonological component of the lexical entry, in which the CV phonotactics of the language is represented hierarchically:

```
Lu:   <>            == Substantif
      <sem>         == Semantique
      <sem glose>   == arbre
      <phon t>      == h
      <phon c>      == "P_l:<>"
      <phon v 1>    == "P_u:<>"
      <phon v 2>    == '.'.
```

The entire theory (not counting additional lexical items), which also defines a hierarchy of marked and unmarked feature values for syllable structure constituents, has almost 200 phonological and prosodic equations, and is too long to be presented in detail here. Two complex theorems which can be derived from the theory are as follows, using the same lexical item “lu” (‘arbre’ = tree):

```
Lu:< phon schema faisceau > = ([ h * . * [ ( [-contin]
  [+sonore] [+reson] [-implos] [-lab] [+cor] [-vel] ) ^
  ( [+sonore] [+haut] [-moyen] [-anterieur] [-labial] )
  ^ . ] ]) .
Lu:< sem schema > = ( arbre = [+concrete] [-humain] ) .
```

The schemata represent feature structures, together with lexical tone (in this case a High tone, “h”). The Dan/Yacouba language also has lateral and nasal prosody, which are defined in the schema. The “.” instances in the value section of the theorem indicate null values for properties which are not represented in this particular lexical item. The first of these is for either the lexical nasal or lateral prosody which is characteristic of this language and the

second is for a second vowel, neither of which occur in this particular lexical item.

3.5 Tone displacement in Kikuyu

An important realisational property of tones and accents is that they may be temporally displaced relative to the tone or stress bearing unit with which they are associated. Kohler has demonstrated this for the accentuation associated with sentence stress in German (Kohler 1991) and Clements and Ford (1979) have shown that Kikuyu (now usually known as Gikuyu; ISO 639.3: kik) has tone displacement relative to the tone-bearing unit: other things being equal (Kikuyu has a complex tone realisation system), the tone occurs on the following syllable. The facts to be described here are:

1. Kikuyu has agglutive verb morphology, leading to long inflected verbs.
2. The first two tones on each verb are the same: there is a word-initial tone spreading (displacement, delay) feature which copies the tone on the first syllable to the second.
3. The following tones spread across morpheme boundaries.

Two examples are shown in Table 1.

Table 1: Kikuyu tone realisation with tone displacement.

Underlying pattern		Surface pattern	Underlying tone sequence		Surface tone sequence
tòmòrÒrírÉ	→	tòmòròrìrÉ	L L L H H	→	L L L L H
tòmárÒrírÉ	→	tòmàrÓrìrÉ	L H L H H	→	L L H L H

Without going into further details at this point, the theorems to be predicted by the Kikuyu theory will be represented using the "°" operator to mean temporal overlap between the tone and the segmental syllable with which it associated on the lexical or the surface phonetic level. From the queries with no initial "d" the underlying lexical tone assignment is inferred, and from the queries with an initial "d" the sequences with tone displacement are inferred:

```
Kikuyu:< we him look_at tense > = [L°to] [L°mo] [L°rOr]
   [H°i] [H°rE] .
Kikuyu:< d we him look_at tense > = [L°to] [L°mo]
   [L°rOr] [L°i] [H°rE] .
Kikuyu:< we them look_at tense > = [L°to] [H°ma] [L°rOr]
   [H°i] [H°rE] .
Kikuyu:< d we them look_at tense > = [L°to] [L°ma]
   [H°rOr] [L°i] [H°rE] .
```

The complete Kikuyu tone displacement theory from which these theorems are inferred is modelled in DATR as follows:

```
% Query-matching node for morpheme composition
Kikuyu: <> == "Displacement" .
% Morpheme lexicon
To: <> == Word              % Abstract morpheme node
 <gloss> == we              % Semantic gloss in English
 <core 1> == to.            % Syllable structure
```

```
Ma: <> == Word
 <gloss> == they/them
 <core 1> == ma
 <tone> == 'H'.
Mo: <> == Word
 <gloss> == him
 <core 1> == mo.
Ror: <> == Word
 <gloss> == 'look at'
 <core 1> == rOr.
Tom: <> == Word
 <gloss> == send
 <core 1> == tom
 <tone> == 'H'.
Ire: <> == Word
 <gloss> == tense
 <core> == rE
 <core 1> == i
 <tone> == 'H'.
% Morpheme, syllable, word templates
Morpheme: <> == Syllable:<1> Syllable:<2>.
Syllable: <> == [ Tone ° Core '] '
 <2> == <"<core>">
 <no_core> == .
Tone: <> == "<tone>"
 <tone> == 'L'
 <1 'H'> == 'H'
 <1 'L'> == 'L'.
Core: <> == "<core>"
 <core> == no_core.
Word: <> == Morpheme Composition
 <tone> == Tone
 <core> == Core.
% Definition of composition with displacement
Composition: <0> == "Displacement:<>"
 <> == <0>
 <d> == "Displacement:<Tone>"
 <'H'> == <d>
 <'L'> == <d>.
Displacement: <> == "Select_nodisp:<>"
 <0> == <>
 <d> == "Select_disp:<>"
 <'H'> == "Select_H:<>"
 <'L'> == "Select_L:<>".
Select_nodisp: <> == Selection
 <disp> == 0.
Select_disp: <> == Selection
 <disp> == d.
Select_H: <> == Selection
 <disp> == 'H'.
Select_L: <> == Selection
```

```
 <disp> == 'L'.
Selection: <> ==
 <we> == "To:<"<disp>">"
 <they> == "Ma:<"<disp>">"
 <them> == <they>
 <him> == "Mo:<"<disp>">"
 <look_at> == "Ror:<"<disp>">"
 <send> == "Tom:<"<disp>">"
 <tense> == "Ire:<"<disp>">".
```

4 Symbolic and numerical phonetic inference of tone terracing

4.1 Symbolic modelling of tone sandhi relations

A generic account of tone sandhi relations can be given for Niger-Congo languages with two lexico-syntactic tones and tone terracing (more accurately: 'pitch terracing'): after a high tone, raise the following tone slightly, and after a low tone lower the following tone slightly. Individual languages are more complex than this and have different mappings, as well as constraints on tone-spreading from syllable to syllable, on tone blocking by certain consonants, on numbers of tones and on more specific contexts for tone raising and lowering. For present purposes, the generic account will be sufficient to illustrate the inference procedure.

The following inference cases have to be accounted for:

1. The empty input.
2. A single tone with no other context.
3. A tone which is raised after a preceding high tone.
4. A tone which is lowered after a preceding low tone.

This specification is realised very straightforwardly one-to-one in DATR default inference notation, including DATR variables:

```
Tone:                          % Single node to be queried
  <>   ==                      % Default null output
  <$a> == $a                   % Copy variable value
  <h $a>  == h RAISE <$a>      % Output and evaluate rest
  <l $a>  == l LOWER <$a>.     % Output and evaluate rest
```

The first equation indicates: an empty input receives no value; any input consisting of a single item (whether 'h' or 'l', represented by the variable "$a") is evaluated as itself; any item preceded by a High tone is subject to a raising operation; any item (whether 'h' or 'l', hence the use of the variable) preceded by a Low tone is subject to a lowering operation. Note that in the present model, raising and lowering are interpreted as dynamic operations applied to the following item, and not as rule-assigned properties of that item, as is the case in conventional phonologies (see Section 4.2).

A common linguistic notation is to prefix '!' to a following 'h' to mean a downstepped high tone. In the SAMPROSA alphabet for prosody (Gibbon 1997) "!" is used with the more general meaning of pitch lowering, and "^" is used with the more general meaning of pitch raising. Using the SAMPROSA

prosody alphabet, the inferred theorems can be given a more standardised appearance (the structure of the theory is not changed):

```
Tone:
  <>   ==
  <$a> == $a
  <h $a>  == h ^ <$a>
  <l $a>  == l ! <$a>.
```

This theory applies almost unchanged to the tone system of Tem (Tchagbale 1984; ISO 639.3: kdh). A sample set of heorems which can be inferred from this theory is the following:

```
Tone:< h h h h > = h ^ h ^ h ^ h .
Tone:< h l h l > = h ^ l ! h ^ l .
Tone:< l h l h > = l ! h ^ l ! h .
Tone:< l l l l > = l ! l ! l ! l .
```

As in the preceding version of the theory, the variables express the generalisation that the raising and lowering operations apply to both low and high tones. In many languages, this generalisation does not hold as it stands, and different raising and lowering operations may apply to the different tones, and also in different contexts. In such cases, it makes sense to spell out the theory without using variables. A maximally spelled-out (variable-free) version of the theory contains seven equations, one for each case of null input, single tone, or pair of tones at the beginning of the tone sequence to be evaluated, as opposed to the four equations of the generalised theory with variables:

```
Tone:
  <> ==
  <h> == h
  <l> == l
  <h h> == h ^ <h>
  <h l> == h ^ <l>
  <l h> == l ! <h>
  <l l> == l ! <l>.
```

If a more traditional modelling convention with the raising and lowering diacritic is required, then still less generalisation is possible, since each separate symbol with diacritic has to be matched separately:

```
Tone:
  <> ==
  <h> == h
  <l> == l
  <^h> == ^h
  <^l> == ^l
  <!h> == !h
  <!l> == !l
  <h h> == h <^h>
  <h l> == h <^l>
  <l h> == l <!h>
  <l l> == l <!l>
  <^h h> == ^h <^h>
  <!h l> == !h <^l>
  <^l h> == ^l <!h>
```

```
  <!l l> == !l <!l>.
```

This theory derives the following theorems from the same input query:

```
Tone:< h h h h > = h ^h ^h ^h .
Tone:< h l h l > = h ^l !h ^l .
Tone:< l h l h > = l !h ^l !h .
Tone:< l l l l > = l !l !l !l .
```

The principle of Occam's Razor (in informal terms: the simpler the better) indicates that the general operation rather than the property diacritic is the better way to go.

However, another way to go is to generalise over the contexts rather than over the operations, as shown by the following theory, which introduces separate nodes with separate DATR local contexts for the two tone contexts, and states that High tones lead to raising and Low tones lead to lowering:

```
Tone:
  <>   ==
  <h>  == h Tone_high:<>
  <l>  == l Tone_low:<>.
Tone_high:
  <>   == Tone
  <h>  == ^h <>
  <l>  == ^l Tone_low:<>.
Tone_low:
  <>   == Tone
  <h>  == !h Tone_high:<>
  <l> == !l <>.
```

This theory generates exactly the same theorems as those generated by the previous theory.

Assuming a language such as Baule (ISO 639.3: bci) in which all four contexts are differentiated (Ahoua 1996), each raising and lowering case can be identified with the names used in the literature: upsweep for High following High, upstep for Low following High, Downstep for High following Low, and Downdrift for Low following Low:

```
Tone:
  <>   ==
  <h>  == high Tone_high:<>
  <l>  == low Tone_low:<>.
Tone_high:
  <>   == Tone
  <h>  == upsweep <>
  <l>  == upstep Tone_low:<>.
Tone_low:
  <>   == Tone
  <h>  == downstep Tone_high:<>
  <l> == downdrift l <>.
```

Theorems inferred from this theory include:

```
Tone:< h h h h > = high upsweep upsweep upsweep .
Tone:< h l h l > = high upstep downstep upstep .
Tone:< l h l h > = low downstep upstep downstep .
Tone:< l l l l > = low downdrift downdrift downdrift.
```

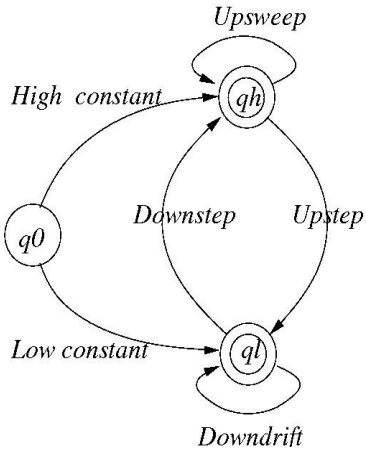

Figure 2: Visualisation of the generic tone theory as a Finite State Transition Network.

An interesting feature of the state generalisation technique is that the structure of the theory now resembles the nodes and transitions of a finite state transition network, as shown in Figure 2 (equivalently the states and transitions of a finite state transducer, or the nonterminal and terminal symbols of a regular grammar, or a regular expression, cf. Gibbon 2001). Figure 2 shows a 'système où tout se tient' following the compositional interpretation paradigm.

4.2 Tone-to-pitch mapping in tone sandhi automata

Traditional phonological models are not directly concerned with physical real-feature and real-time mappings. From a modern perspective, in which measurement techniques and instruments are widely available and in which speech technology systems use sophisticated linguistic input, whether by rule or harvested from annotated data, this is an increasingly inadequate perspective.

The default inference based tone sequencing automaton described in the preceding subsection can easily be modelled within the same theory structure using the numerical operations of the ZDATR inference engine in order to generate actual pitch hypotheses. Any other programming language can be used to generate such sequences; the point of using ZDATR is to provide a homogeneous modelling environment rather than a linguistic notation coupled with an ad hoc implementation. The ZDATR model can be converted into a 'production system' at a later stage.

The modelling convention on which the illustration of numerical phonetic mapping is based in the present theory is overly simple:

$pitch_i = pitch_{i-1} * z$, where i is the position of the tone and z varies according to automaton context.

A more adequate model additionally has (at least) a baseline component and an interpolation function, but the principle is the same, for example:

$pitch_i = baseline + (pitch_{i-1} - baseline) * z$, where the baseline may also have an independent declination function.

A modified node based tone sequencing theory is shown here:

```
Tone:
  <>    ==
  <$a>    == $a
  <$a h>   == Tone_high:<Multiply:<$a '1.1' .>>
  <$a l>   == Tone_low:<$a>.
Tone_high:
  <>    == Tone
  <$a h>   == $a <Multiply:<$a '1.01' .>>
  <$a l>   == $a Tone_low:<Multiply:<$a '0.90' .>>.
Tone_low:
  <>    == Tone
  <$a h>   == $a Tone_high:<Multiply:<$a '1.05' .>>
  <$a l>   == $a <Multiply:<$a '0.95' .>>.
```

The inferred pitch values for the tone sequences HHH, HLHL, LHLH and LLLL, given an onset pitch seed value for initial Low tones (in this case 120 Hz) which is required in order to calculate the absolute pitch levels of the output, are contained in the following theorems:

```
Tone:< 120 h h h h > = 132 133.32 134.653 136 .
Tone:< 120 h l h l > = 132 118.8 124.74 112.266 .
Tone:< 120 l h l h > = 120 126 113.4 119.07 .
Tone:< 120 l l l l > = 120 114 108.3 102.885 .
```

The sequence shows the desired result: the pitch difference between H and L is larger than the pitch difference between L and H, leading to the 'terracing effect' which can be observed for HLHL and LHLH in Figure 3.

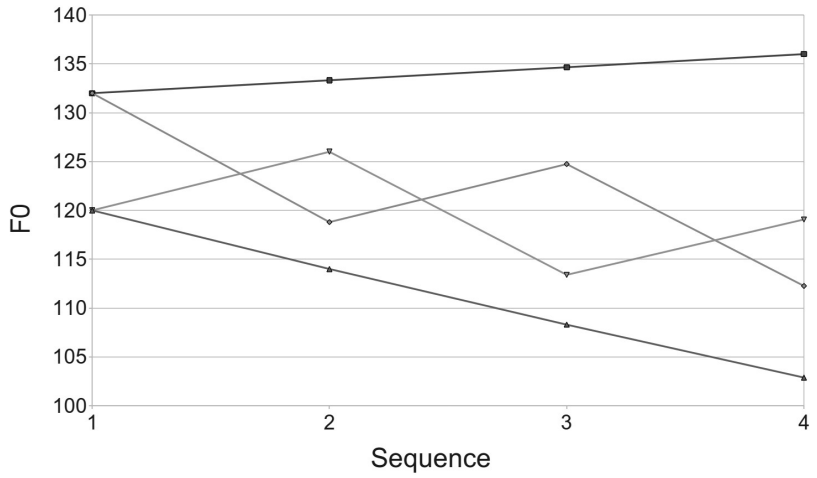

Figure 3: Pitch sequence output for pitch onset of 120 Hz and sequences HHHH, HLHL, LHLH and LLLL.

5 The ZDATR inference engine: background and performance

The ZDATR inference engine implementation (Gibbon & Strokin 1998) is available in both open source and free binary distributions (libraries may have to be adapted, depending on local operating system distributions).[6] A web interface, ZDATRweb, is provided for demonstrating and testing DATR theories using the ZDATR inference engine. ZDATR was developed as an efficient, industry standard command-line controlled and pipeline-capable implementation of DATR in C for Unix flavours (Linux, Solaris) and for MS-Windows, and has been widely used for applications in phonology, prosody, morphology and semantics. ZDATRweb was developed for the rapid testing of complex inferences and for teaching purposes, and contains two interfaces (stable since 1996): the *ZDATR Testbed*,[7] with pre-installed DATR theories, and the *ZDATR Scratchpad*[8] for testing small theories (see Figure 4 for screenshots of one of the tone theories with the ZDATR scratchpad).

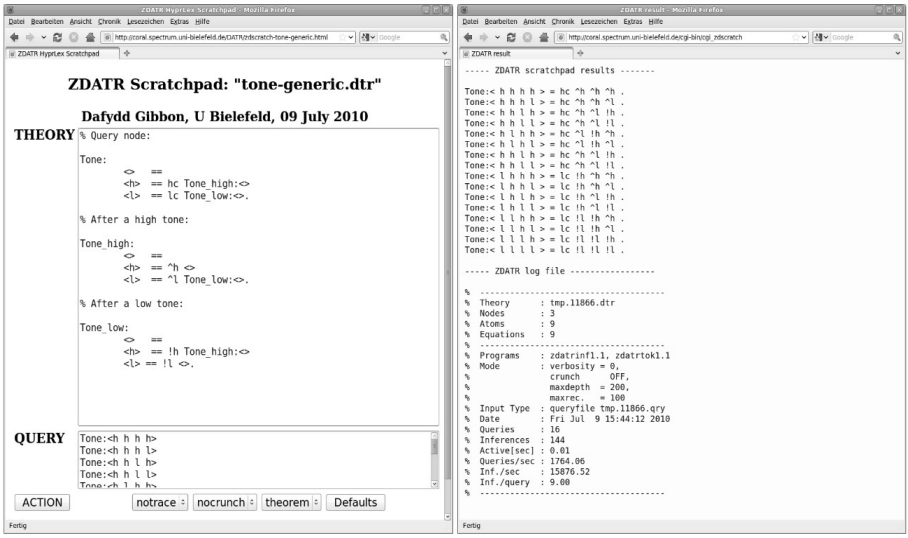

Figure 4: Screenshots of ZDATR Scratchpad: theory and queries (left), inferred theorems and logfile (right).

The performance of the ZDATR inference engine is logged for both the compiler and the inference engine. Compiler speed is theoretically linear in the length of the theory since the compiler is essentially a deterministic finite state tokeniser, and inference engine speed is theoretically quadratic in the length of the input since the inference engine is a deterministic recursive tree processor. However, the theories discussed in the present contribution are too short to demonstrate their complexity empirically.

6 http://www.spectrum.uni-bielefeld.de/DATR/Zdatr/
7 http://www.spectrum.uni-bielefeld.de/DATR/zdtestbed.html
8 http://www.spectrum.uni-bielefeld.de/DATR/zdscratch.html

Nevertheless, the log files for the inference engine give indirect (albeit somewhat spurious) information about the efficiency of the engine. The reason for this is that the timing information in the log files is heavily influenced by unknown properties of the operating system, particularly applications which take so little time. The times include file operations, which in turn presumably involve unpredictable caching operations. The log files for four of the theories discussed above are shown below for comparison. In each case, the shortest time out of 5 runs was taken in order to minimise operating system effects.

```
-----------------------------------
Theory      : tones-generic.dtr
Nodes       : 94
Atoms       : 137
Equations   : 223
Date        : Mon Jul  5 09:33:49 2010
Queries     : 31
Inferences  : 227
Queries/sec : 7133.00
Inf./sec    : 52231.94
Inf./query  : 7.32
-----------------------------------
Theory      : tones-numeric.dtr
Nodes       : 155
Atoms       : 136
Equations   : 285
Date        : Mon Jul  5 09:33:56 2010
Queries     : 31
Inferences  : 642
Queries/sec : 4836.95
Inf./sec    : 100171.63
Inf./query  : 20.71
-----------------------------------
Theory      : tones-onenode-novars.dtr
Nodes       : 94
Atoms       : 136
Equations   : 240
Date        : Mon Jul  5 09:34:11 2010
Queries     : 31
Inferences  : 235
Queries/sec : 7085.71
Inf./sec    : 53714.29
Inf./query  : 7.58
-----------------------------------
Theory      : tones-onenode-vars.dtr
Nodes       : 94
Atoms       : 136
Equations   : 240
Date        : Mon Jul  5 09:34:05 2010
Queries     : 31
Inferences  : 235
Queries/sec : 7043.85
```

```
Inf./sec    : 53396.96
Inf./query  : 7.58
-----------------------------------
```

The node, atom and equations counts refer to the numbers of instances in the inference process, not to the number of occurrences in the theory:

1. 94 node instances, derived from the 3 nodes of the theory multiplied by the 31 shown queries plus 1 query node: 31*3+1=94.
2. 136 atom instances (137 in the numerical theory).
3. 240 equation instances.

As already noted, the timing results on such short theories are not very helpful, since they extrapolate to far larger numbers per second than are justified by the actual fractions of a second required by the inference procedure. However, a number of tendencies can be tentatively summarised as follows:

1. Complexity:
    1. The three symbolic theories do not differ noticeably in complexity in the compiled model, each involving approximately the same number of inferences (around 230) and inferences per query (around 8, i.e. 230/31) despite differences in structure.
    2. The numerical theory is more complex (by a factor of 3) than the symbolic theory in the compiled model, with more inferences (642) and inferences per query (21).
2. Timing:
    1. The symbolic theories have approximately the same timing (around 7k queries/s and 53k inferences/s).
    2. The numerical theory is noticeably slower with queries than the symbolic theories (around 5k queries/s), but has many more inferences to process in this time (around 100k inferences/s).

The main conclusion to be drawn is that for linguistic modelling purposes, the ZDATR engine is very suitable. Before query times of more than 1 second per query can be expected, the theory would have to be many orders of magnitude larger (or the queries many orders of magnitude longer).

6 Conclusion

The aim of the present contribution was to demonstrate the amenability of prosodic problems to explication with a highly general inference strategy taken from default logic as represented by the DATR formalism, and implemented with an efficient inference engine, ZDATR, as a theorem prover. A number of prosodic problems were discussed: compound stress in German, nuclear tone structures in English, lexical tone in Yacouba, tone displacement in Kikuyu, and the symbolic and numerical phonetic interpretation tone sequences in Niger-Congo terraced tone languages.

In each of these cases, the correct theorems are predicted using the default logic inference strategy. The graphical visualisations of the theory structures further demonstrate the criterion of connectedness for linguistic theories with the aim of explaining the structure of language as “un système où tout se tient”,

a fully connected system. As in other computationally modelled theories of language, strict criteria of consistency, connectedness, completeness, soundness and precision are fulfilled by the theories presented here.

The value of this approach is not purely theoretical or metatheoretical. Several of these techniques have been used in large-scale speech technology applications: the German stress assignment strategy was used in lexicon construction in the Verbmobil project of the 1990s (Gibbon & Bleiching 1991; Bleiching 1992, 1994), the tone sequence realisation strategy has been used in text-to-speech synthesis of West African languages (Gibbon & Urua 2006), and related techniques were used by Andry & al. (1992) in a speech recognition lexicon application.

Bibliography

Ahoua, Firmin (1996). *Prosodic aspects of Baule: with special reference to the German of Baule speakers*. Köln: Köppe-Verlag.

Andry, Francois, Norman Fraser, Scott McGlashan, Simon Thornton & Nick Youd (1992). Making DATR work for speech: lexicon compilation in SUNDIAL. *Computational Linguistics* 18. 245-267.

Bleiching, Doris (1992). Prosodisches Wissen im Lexikon. In: G. Görz, ed., *KONVENS 92.* Berlin: Springer-Verlag, pp.59-68.

Bleiching, Doris (1994). Integration von Morphophonologie und Prosodie in ein hierarchisches Lexicon. In Harald Trost, ed., *Proceedings of KONVENS-94*, pp. 32-41, Vienna: Oesterreichische Gesellschaft fuer Artificial Intelligence.

Barg, Petra, James Kilbury, Ingrid Renz, Christof Rumpf & Sebastian Varges (2005). *QDATR Manual Version 4.0.*
`http://www.phil-fak.uni-duesseldorf.de/sfb282/B3/qdatr.html`

Clements, George N. & Kevin C. Ford (1979). Kikuyu tone shift and its synchronic consequences. *Linguistic Inquiry* 10:2, 179-210.

Evans, Roger & Gerald Gazdar (1996). DATR: A language for lexical knowledge representation. In: *Computational Linguistics* 22:2, 167-216.

Evans, Roger & Gerald Gazdar (1998). *The DATR Standard Library RFC, Version 2.00*. University of Sussex.

Finkel, Raphael, Lei Shen, Greg Stump and Suresh Thesayi (n.d.). KATR: A Set-Based Extension of DATR. University of Kentucky, Technical Report No. 346-02.

Gibbon, Dafydd (1976). *Perspectives of Intonation Analysis*. Bern etc.: Lang.

Gibbon, Dafydd (1992). ILEX: A linguistic approach to computational lexicology. In: U. Klenk, ed., *Computatio Linguae*, Beiheft zur *Zeitschrift für Dialektologie und Linguistik*. Stuttgart: Steiner. 32-53.

Gibbon, Dafydd (2001). Finite state prosodic analysis of African corpus resources. *Proceedings of Eurospeech* 2001, Aalborg, Denmark, pp. 83-86.

Gibbon, Dafydd, Roger Moore & Richard Winski (1997). *Handbook of Standards and Resources for Spoken Language Systems*. Berlin: Mouton de Gruyter.

Gibbon, Dafydd & Firmin Ahoua (1991). DDATR: un logiciel de traitement d'héritage par défaut pour la modélisation lexicale. In: *Cahiers Ivoiriens de Recherche Linguistique* (CIRL) 27:5-59.

Gibbon, Dafydd & Doris Bleiching (1991). An ILEX model for German compound stress in DATR. *Proceedings of the FORWISS-ASL Workshop on Prosody in Man-Machine Communication*, pp. 1-6.

Gibbon, Dafydd & Grigory Strokin (1998). *ZDATR Version 2.0 Manual*. Universität Bielefeld.

Gibbon, Dafydd, Eno-Abasi Urua (2006). Morphotonology for TTS in Niger-Congo languages. Proceedings of Speech Prosody 2006. Dresden, Germany.

Kilbury, James, Wiebke Petersen, Christof Rumpf (2006). Inheritance-based models of the lexicon. In: Dieter Wunderlich (ed.), *Advances in the Theory of the Lexicon.* Berlin/New York: Mouton de Gruyter, pp. 429-477.

Kohler, K. J. (1991). Terminal intonation patterns in single-accent utterances of German: Phonetics, phonology and semantics. In Klaus J. Kohler, ed., *Studies in German Intonation*, AIPUK 25, Kohler, K. J. (ed.), pp. 115-187.

Tchagbale, Z. 1984. T.D. de Linguistique: exercises et corrigés. Institut de Linguistique Appliquée, Université Nationale de Côte d'Ivoire, Abidjan, No. 103.